SCHEDAE INFORMATICAE
MCCLIX VOLUME 11 2002

A Neural Economic Time Series Prediction with the Use of a Wavelet Analysis

Paweł Hajto
Institute of Computer Science, Jagiellonian University
e-mail: hajto@elf.ii.uj.edu.pl

Abstract. In this article a wavelet and artificial neural networks theory is used to predict economic time series in a described computer application. Its predicting capabilities were tested on a USD/PLN average exchange ratio and discussed in this paper. The achieved results are satisfactory.

1. Introduction

The main theme of this article is the mathematical theory needed to develop a computer application that helps to predict economic data, which contain an element of time. This is the case of stock markets, currency exchange rates, inflation rates, etc. In the second part of this paper an example application and its forecasting results are described. The application was used to predict USD/PLN average exchange rates.

Financial forecasting is undoubtedly the most advanced artificial neural networks application in economical sciences. There are many references concerning problems of stock, currency, debentures market processes' analysis [2, 4, 5, 19, 22, 24, 25, 28].

A similar prediction problem is covered in D. Witkowska's book [26]. The author discusses a neural model and statistical methods used to forecast an inflation ratio. The model consists of one neural network, with no wavelet transforms applied to input data. The average percentage error of predicted values is 0,95–3,68% ([26], p. 116), depending on details of the neural model

used. The application described in this paper achieves an average percentage error of 0,54–1,22%.

This work is inspired with P. Lula's book [14], where several economical applications of artificial neural networks are covered. Lula conducts an empirical verification of the market efficiency hypothesis, using a neural-wavelet model and basing on Warsaw Stock Exchange index data, achieving some very interesting results, although not confirming the hypothesis ([14], pp. 156–164). However, he states that this model may lead to prognosis of practical usefulness and that Discrete Wavelet Transform is an adequate tool for time series analysis.

2. Discrete wavelet transform

The wavelet theory evolved in mid-eighties of the past century ([3, 12, 16, 21]), though some constructions and theoretical results were discovered much earlier ([6, 7, 20, 21]). It can be regarded as Fourier analysis extension, specially in the scope of signal processing. Wavelets are functions, whose localizations in time and frequency can be fully controlled. This leads to improved and new signal processing applications. Wavelet transforms are used in physics, geophysics, astronomy, biology, chemistry, image processing (NMR, tomography), sound processing, data compression and – economics.

2.1. Basic facts from the wavelet theory

Definition 1. A function $\Psi(t) \in L_2(\mathbb{R})$ is a *wavelet*, if the functions

$$\Psi_{j,k} := 2^{\frac{j}{2}}\Psi(2^j t - k), \quad j,k \in \mathbb{Z}$$

create an orthonormal basis in $L_2(\mathbb{R})$, where $L_2(\mathbb{R})$ denotes the set of functions $f : \mathbb{R} \to \mathbb{C}$, such that:

$$\int_{-\infty}^{\infty} |f(t)|^2 dt < \infty$$

with the inner product defined by:

$$f \circ g = \int_{-\infty}^{\infty} f(t)\overline{g(t)}dt.$$

An example is the Haar wavelet, defined as follows:

$$\Psi := \begin{cases} 1 & \text{for } t \in [0, \frac{1}{2}), \\ -1 & \text{for } t \in [\frac{1}{2}, 1], \\ 0 & \text{otherwise.} \end{cases}$$

DEFINITION 2. A *multiresolution analysis (MRA)* is a nested sequence

$$\ldots \subset V_{-1} \subset V_0 \subset V_1 \subset \ldots$$

of subspaces of $L_2(\mathbb{R})$ satisfying

1. $\bigcup_{n \in Z} V_n$ is dense in $L_2(\mathbb{R})$,
2. $\bigcap_{n \in Z} V_n = \{0\}$,
3. $f(t) \in V_n$ if and only if $f(2^{-n}t) \in V_0$,
4. there exists a function $\Phi(t)$, called *a scaling function*, such that $\{\Phi(t-k)\}_{k \in \mathbb{Z}}$ is an orthonormal basis for V_0.

FACT 1. *Because $\Phi \in V_0 \subset V_1$, condition 3 of MRA definition implies, that $\Phi(x/2) \in V_0$. This leads to*

$$\Phi(x/2) = \sum_{n \in \mathbb{Z}} a_n \Phi(x-n).$$

We define m_Φ:

$$m_\Phi(\xi) = \frac{1}{2} \sum_{n \in \mathbb{Z}} a_n e^{-in\xi}.$$

There exists a relationship between wavelets and a multiresolution analysis ([27], p.45):

THEOREM 1. *Let us suppose, that we have a MRA. A function $\Psi \in W_0 = V_1 \ominus V_0$ is a wavelet if and only if*

$$\hat{\Phi}(\xi/2) = e^{i\xi/2} v(\xi) \overline{m_\Phi}(\xi/2 + \pi) \hat{\Psi}(\xi/2),$$

where $\hat{\Phi}$ and $\hat{\Psi}$ are Fourier transforms of Φ and Ψ respectively, $v(\xi)$ is a 2π-periodic function such that $|v(\xi)| = 1$.

Additionally, for Ψ and every $s \in \mathbb{Z}$ span$\{\psi_{j,k}\}_{k \in \mathbb{Z}, j<s} = V_s$.

If $v = 1$, the wavelet Ψ is defined by:

$$\Psi(x) = \sum_{n \in \mathbb{Z}} \overline{a_n} (-1)^n \Phi(2x + n + 1),$$

where $a_n = \int_{-\infty}^{\infty} \Phi(x/2)\overline{\Phi}(x-n)dx$.

DEFINITION 3. Having a MRA, we define an *orthogonal* subspace $V_j^{\perp} \subset L_2(\mathbb{R})$ to subspace $V_j \subset L_2(\mathbb{R})$ with the following condition:

$$V_j \oplus V_j^{\perp} = V_{j+1}.$$

The MRA definition implies (see [27], p. 41) that

$$L_2(\mathbb{R}) = \oplus \sum_{j \in \mathbb{Z}} V_j^{\perp}.$$

The theory of a multiresolution analysis states that if a MRA is given, we can find a function Ψ, which generates an orthonormal wavelet basis for V_s for all $s \in \mathbb{Z}$, in other words, $\text{span}\{\psi_{j,k}\}_{k \in \mathbb{Z}, j<s} = V_s$. In practical applications we are interested in examining the orthogonal projections $P_n(f)$ of a function $f \in L_2(\mathbb{R})$ onto wavelet spaces V_n. This process is realized by using *wavelet filters* (see [1], p. 70, [10] 7.1–7.8).

2.2. Signal processing by wavelets

A given signal $s = [\ldots, s_{-1}, s_0, s_1, \ldots]$ defines a function $f \in V_n$ by

$$f = \sum_{k \in \mathbb{Z}} s_k \psi_{k,n}. \tag{1}$$

Now the wavelet filters process this signal by using two operators, H (the low-pass filter) and G (the high-pass filter), where

$$H(s)_k = \sum_{j \in \mathbb{Z}} h_{j-2k} s_j$$

and

$$G(s)_k = \sum_{j \in \mathbb{Z}} g_{j-2k} s_j.$$

The sequences $\{h_k\}$, $\{g_k\}$ arise from MRA and inner product properties (see [1], p. 70) and are unique for every wavelet family.

Having a signal s, and the associated function $f \in V_n$ (as in 1), $H(s)$ are coefficients of the orthogonal projection $P_{n-1}(f)$ onto V_{n-1} and $G(s)$ coefficients of $P_{n-1}(f)$ onto $V_{n-1}^{\perp}$. A good practical interpretation of this is that $H(s)$ and $G(s)$ contain the low and the high frequencies respectively.

Once we know how to decompose a signal s, it is equally important to have a tool to recompose it. Each of the operators H and G has a so-called dual operator, denoted H^* and G^* respectively, defined by

$$H^*(s^*)_k = \sum_{j\in\mathbb{Z}} h_{k-2j} s^*_j$$

and

$$G^*(d^*)_k = \sum_{j\in\mathbb{Z}} g_{k-2j} d^*_j.$$

The filters and their dual operators act as follows

$$s = H^*(H(s)) + G^*(G(s)).$$

In real world we cannot deal with sequences of the infinite length. The wavelet families that are used (Daubechies, CDF, etc.) have a finite number of non-zero $\{h_k\}$, $\{g_k\}$ filter coefficients. And the solutions for the assumption of the infinite length of the signal s are periodization, mirroring, Gram-Schmidt boundary filters and zero-padding (see [10], Section 10).

2.3. Mallat's pyramid algorithm

The algorithm for processing a signal using wavelet filters is called *a Mallat's pyramid algorithm.*

Let us consider a finite signal $s = [s_0, s_1, \ldots, s_{2^n-1}]$, and wavelet filters H, G with $\{h_k\}$, $\{g_k\}$ coefficients from a chosen wavelet family. Frequencies in s range from 0 to f_N, where f_N is the Nyquist frequency, the highest frequency one can observe in a signal sampled with sampling frequency f_S, $f_N = \frac{f_S}{2}$.

We compute $s^1 = H(s)$ and $d^1 = G(s)$. The length of s^1, d^1 is 2^{n-1} (see [1], p.72). The frequencies contained in s^1 range from 0 to $\frac{f_N}{2}$ (the low part) while in d^1 from $\frac{f_N}{2}$ to f_N (the high part).

Then we apply the same procedure to s^1, obtaining s^2 and d^2, each of length 2^{n-2}. The available frequencies are: 0 to $\frac{f_N}{4}$ (s^2) and $\frac{f_N}{4}$ to $\frac{f_N}{2}$ (d^2).

After n steps the algorithm stops and we get a vector

$$s^* = [s^n_0, d^n_0, d^{n-1}_0, d^{n-1}_1, \ldots, d^2_{2^{n-2}-1}, d^1_0, \ldots, d^1_{2^{n-1}-1}].$$

This is the discrete wavelet transform (DWT) of s. To this form of s one can apply some operations like zero-padding of high-frequency coefficients for noise reduction or to separate only the desirable frequencies in order to get data to train an ANN, which was important in the described application.

Obviously an inverse process is also possible, using H^* and G^* operators and a reversed version of the Mallat's algorithm. It is called the inverse discrete wavelet transform (iDWT).

3. Artificial neural networks

In recent years artificial neural networks (ANNs) have been a topic of very intensive research. A lot of papers have been devoted to various ANNs applications, like speech and pattern recognition, robotics, expert systems, control theory. A large number of applications is present [8, 15, 18, 14, 17, 23, 26, 29].

ANNs are eagerly used because of their properties to approximate non-linear functions and good generalization abilities that help to predict data not included in the learning patterns.

3.1. Basic definitions

DEFINITION 4. A *neuron* is a function

$$F : X \ni x \longmapsto f(w \circ x) \in \mathbb{R},$$

where:

1. X is a set of signals, $X \subset \mathbb{R}^k$,
2. $w \in \mathbb{R}^k$ is a *vector of weights*,
3. $x \in X$ is a signal,
4. $g : \mathbb{R} \to \mathbb{R}$ is an *activation function*.

In the described application the logistic activation function $g(x) := \frac{1}{1+exp(-x)}$ is used.

DEFINITION 5. A *layer of neurons* is a vector function

$$L : X \ni x \longmapsto [F_1(w_1, x), F_2(w_2, x), \ldots, F_l(w_l, x)] \in \mathbb{R}^l,$$

where:

1. X is a set of signals, $X \in \mathbb{R}^k$,
2. F_i, $i = 1 \ldots l$ are the layer's neurons,
3. w_i, $i = 1 \ldots l$ are their vectors of weights.

DEFINITION 6. Given a set of layers, $L_1, \ldots, L_n$, satisfying:

1. $L_1 : X_1 \to X_2$, $X_1 \subset \mathbb{R}^{k_1}$, $X_2 \subset \mathbb{R}^{k_2}$

2. $L_2 : X_2 \to X_3$, $X_3 \subset \mathbb{R}^{k_3}$

3. ...

4. $L_n : X_n \to X_{n+1}$, $X_{n+1} \subset \mathbb{R}^{k_{n+1}}$

we define a *feed-forward, multilayer neural network* as a function

$$N : X_1 \ni x \longmapsto y = L_n(L_{n-1}(\dots L_1(x)\dots)) \in X_{n+1}.$$

This kind of ANN is also called *a Multilayer Perceptron (MLP).*

All ANNs that are in the scope of this article are MLPs, because they are frequently used as universal approximating functions.

3.2. The process of learning

The problem of training an MLP looks as follows. We have a set of n pairs $\{(x_i, y_i)\}_{i=1..n}$, where $x_i \in \mathbb{R}^k$, $y_i \in \mathbb{R}^l$. The pairs are called *patterns*, x_i is the *input pattern*, y_i the network's expected output. We expect the MLP $N : \mathbb{R}^k \to \mathbb{R}^l$ to realize a mapping $N(x_i) = y_i$, $i = 1 \dots n$. A typical learning algorithm consists of estimating errors $\epsilon_i = \|y_i - \overline{y}_i\|$, $i = 1 \dots n$, then $\delta^{(i)}_{k,j} = \delta^{(i)}_{k,j}(\epsilon_i)$ and changing the weights $w^{new}_{k,j} := w^{old}_{k,j} + \delta^{(i)}_{k,j}$, Where $w_{k,j}$ means the j-th entry in the vector of weights of the k-th neuron of network N. The process stops as the total error $\epsilon = \sum_{i=1}^{n} \epsilon_i$ is *small enough.*

The quality of an ANN and of its learning process is not necessarily the value of final error at the end of training. We could use a well known approximation or interpolation method from numerical analysis to find a function realizing the x_i to y_i, $i = 1 \dots n$ mapping. What we expect from an ANN is a good *generalization.* In order to test this, another set of patterns is created, $\{(x^{test}_i, y^{test}_i)\}_{i=1..m}$, but they do not take part in the learning process. Instead, one can check the ability of generalization of an MLP, computing the total error $\epsilon^{test} = \sum_{i=1}^{n} \|N(x^{test}_i) - y^{test}_i\|$. That is a good ANN's quality measure.

An important fact from the ANNs theory, is that the theorem of Hecht-Nielsen (see [9]) states that for a given continuous function $f : \mathbb{R}^k \to \mathbb{R}$ and an awaited approximation error ϵ there always exists a three-layer (input layer, hidden layer and output layer) MLP, which approximates the function. The approximation error of this MLP is below ϵ. The theorem does not describe the activation functions needed for this MLP.

In recent years ANNs have been a topic of an intensive study. Many training algorithms have been developed, like backpropagation, quickpropagation, genetic algorithms methods and others (see [8, 15, 17, 18, 23, 29]). However, an exact description of these algorithms is unnecessary in the scope of this paper. We want to focus on a practical ANN application.

4. ANNs, wavelets and economic time series

In an application of economic time series prediction a typical set of data is a signal $x = [x_1, \ldots, x_k]$, containing e.g. stock market index values or currency exchange rates. Each entry comes from another time point, which means, that x_1 is the exchange rate value at the beginning, x_2 the value on the next day and so on.

4.1. The basic approach

The problem of value forecasting can be defined in the following way. Having the values $x_1, \ldots, x_k$ of economic data at consecutive time points (e.g. stock rates at day no. 1, day no. 2, day no. 3) it is desired to estimate its unknown value on the forthcoming day. In other words, we would like to have a function $\dot{f} : \mathbb{R}^k \to \mathbb{R}$, such as:

$$x_{k+1} = \dot{f}(x_1, \ldots, x_k),$$

where x_{k+1} is the expected unknown value.

Obviously, the future is generally unpredictable. And to find such a function $\dot{f}$ is not possible. What can be done, to tray to observe as much regularity of our data as possible and to look for a function $f : \mathbb{R}^k \to \mathbb{R}$, such that:

$$\tilde{x}_{k+1} = f(x_1, \ldots, x_k)$$

where the distance $|x_{k+1} - \tilde{x}_{k+1}|$ is small enough.

A first idea could be splitting our data into patterns for an MLP. Assuming the data is a vector $x = [x_1, x_2, ..., x_m]$ the patterns look like this:

$$\begin{array}{l} ([x_1, x_2, ..., x_k], x_{k+1}) \\ ([x_2, x_3, ..., x_{k+1}], x_{k+2}) \\ ([x_3, x_4, ..., x_{k+2}], x_{k+3}) \\ \ldots \\ ([x_{m-k}, x_{m-k+1}, ..., x_{m-1}], x_m), \end{array}$$

where $k < m$.

Next, an architecture for an ANN must be chosen. Lula designed a network ([14], p. 158) for testing the market efficiency hypothesis basing on Warsaw Stock Exchange index data. The author uses an MLP with three layers, 6 neurons in the input layer, 6 neurons with a tangensoidal activation function and 1 neuron in the output layer with a linear activation function. The value of $k = 6$ is estimated with a BDS input data test, described in [13].

After this MLP is trained, it realizes the function f for $k = 6$

$$\begin{aligned}
&x_7 \approx f(x_1, \ldots, x_6)\\
&x_8 \approx f(x_2, \ldots, x_7)\\
&x_9 \approx f(x_3, \ldots, x_8)\\
&\ldots\\
&x_m \approx f(x_{m-6}, x_{m-5}, ..., x_{m-1}).
\end{aligned}$$

That is just an input (known) data approximation. But now we can try to estimate the *unknown* values:

$$\begin{aligned}
&\tilde{x}_{m+1} = f(x_{m-5}, x_{m-4}, \ldots, x_m)\\
&\tilde{x}_{m+2} = f(x_{m-4}, x_{m-3}, \ldots, x_m, \tilde{x}_{m+1})\\
&\tilde{x}_{m+3} = f(x_{m-3}, x_{m-2}, x_{m-1}, x_m, \tilde{x}_{m+1}, \tilde{x}_{m+2}).\\
&\ldots
\end{aligned}$$

However, this basic "one-network" idea has not been used in this work, because of the poor results Lula achieved with the Warsaw Stock Exchange index. Despite using sophisticated training algorithms the DIR coefficient (the part of correctly guessed directions of fluctuations) on testing patterns was only 61% ([14], p. 159). This results are of low practical usefulness. The MLPs used in an application described in this paper achieved a DIR on testing patterns of ca. 86%–90%. But the patterns contained wavelet filtered oscillations, not raw economic data.

4.2. The wavelet approach

The wavelet approach bases on applying the Mallat's pyramid algorithm to the given data, splitting the data into separated frequency bands, approximating each band by an ANN and predicting their values as described above.

The input data is a vector $x = [x_0, \ldots, x_{2^n-1}]$. The assumption of its length is important because of the Mallat's algorithm. In practical applications zero-padding can be used to achieve this.

We compute the DWT of x, getting a vector

$$x^* = [x_0^n, d_0^n, d_0^{n-1}, d_1^{n-1}, \ldots, d_{2^{n-2}-1}^2, d_0^1, \ldots, d_{2^{n-1}-1}^1].$$

In order to split x into different frequency ranges we need to set all entries in x^* responsible for unwanted frequencies to zero.

Range	Vector
$\frac{f_N}{2}$ to f_N	$x^{(n)*} = [0, \ldots, 0, d_0^1, \ldots, d_{2^{n-1}-1}^1]$
$\frac{f_N}{4}$ to $\frac{f_N}{2}$	$x^{(n-1)*} = [0, \ldots, 0, d_0^2, \ldots, d_{2^{n-2}-1}^2, 0, \ldots, 0]$
...	...
$\frac{f_N}{2^n}$ to $\frac{f_N}{2^{n-1}}$	$x^{(1)*} = [0, d_0^n, 0, \ldots, 0]$
0 to $\frac{f_N}{2^n}$	$x^{(0)*} = [x_0^n, 0, \ldots, 0]$.

Now the inverse DWT of each $x^{(i)*}$ is computed:

$$y^{(i)} = IDWT(x^{(i)*}),$$

where $i = 0 \ldots n$.

Note that $y^{(i)}$ contains a range of frequencies from x as shown above and its length is 2^n.

To approximate and predict $y^{(i)}$ for $i = 1 \ldots n$ MLPs are used with the same three layer architecture as shown in the basic approach. The patterns are in the form:

$$([y_1^{(i)}, y_2^{(i)}, ..., y_6^{(i)}], y_7^{(i)})$$

$$([y_2^{(i)}, y_3^{(i)}, ..., y_7^{(i)}], y_8^{(i)})$$

$$([y_3^{(i)}, y_4^{(i)}, ..., y_8^{(i)}], y_9^{(i)})$$

$$\ldots$$

$$([y_{n-6}^{(i)}, y_{n-5}^{(i)}, ..., y_{n-1}^{(i)}], y_n^{(i)}),$$

where $i = 1 \ldots n$.

There is no need to build an ANN to approximate $y^{(0)}$ since all the entries in this vector are equal to the mean value of $x_0, \ldots, x_{2^n-1}$.

Let $N^{(i)}$ denote the ANN used to approximate $y^{(i)}$. Unknown values of $y^{(i)}$ can be predicted:

$$\begin{aligned}
&\tilde{y}_{n+1}^{(i)} = N^{(i)}(y_{n-5}^{(i)}, y_{n-4}^{(i)}, \ldots, y_n^{(i)}) \\
&\tilde{y}_{n+2}^{(i)} = N^{(i)}(y_{n-4}^{(i)}, y_{n-3}^{(i)}, \ldots, y_n^{(i)}, \tilde{y}_{n+1})^{(i)} \\
&\tilde{y}_{n+3}^{(i)} = N^{(i)}(y_{n-3}^{(i)}, y_{n-2}^{(i)}, y_{n-1}^{(i)}, y_n^{(i)}, \tilde{y}_{n+1}^{(i)}, \tilde{y}_{n+2}^{(i)}), \\
&\ldots
\end{aligned}$$

where $i = 1 \ldots n$.

Thus

$$\tilde{x}_{n+j} = \sum_{i=1}^{n} \tilde{y}_{n+j}^{(i)} + M,$$

where $j > 0$ and $M = y_0^{(0)}$ is the average value of $x_0, \ldots, x_{2^n-1}$. This is a consequence of wavelet filter properties and the Orthogonal Decomposition Theorem ([1], p. 101).

4.3. A small improvement

There exists a simple method of improving the wavelet-neural prediction. It can be easy observed, that there is no need to approximate low frequency ranges with ANNs if it is intended to forecast just a few values.

In the example of the later discussed application a data of length 1561 samples was used. It was intended to predict just the next 5 samples. The data was zero padded to achieve a length of 2^{11} and split into 11 frequency ranges.

Range	Oscillations length
$\frac{f_N}{2}$ to f_N	2-4 samples
$\frac{f_N}{4}$ to $\frac{f_N}{2}$	4-8 samples
...	...
$\frac{f_N}{2^{11}}$ to $\frac{f_N}{2^{10}}$	2048-4096 samples.

Let us denote with $s = [s_1, s_2, \ldots, s_{1561}, 0, \ldots, 0]$ the first data set of length 2^{11} and with $t = [s_1, s_2, \ldots, s_{1561}, s_{1562}, \ldots, s_{1566}, 0, \ldots, 0]$ the other, where $s_{1562}, \ldots, s_{1566}$ are the desired real, not forecasted values.

As there is no way for this future 5 entries $s_{1562}, \ldots, s_{1566}$ to generate long oscillations (i.e. 1024–2048, 512–1024, ..., 64–128 samples) they have very little or no effect on low and medium frequency wavelet coefficients.

So having the $s^* = DWT(s)$ only high frequency bands are separated and used as samples for ANNs (as above). The high frequency coefficients in s^* are zero padded and the $IDWT$ is applied. The resulting signal $\tilde{s}$ is a rough approximation of s *and* of t.

The unknown values $s_{1562}, \ldots, s_{1566}$ are approximated in the following way:

$$s_{1561+j} \approx \sum_{i=k}^{n} \tilde{y}^{(i)}_{1561+j} + \tilde{s}_{1561+j},$$

where in the described application $j = 1, \ldots, 5$, $n = 11$ (the number of frequency ranges). Ranges $k, k+1, \ldots, n$ are approximated by ANNs ($\tilde{y}^{(i)}_{1561+j}$) and $1, \ldots, k-1$ are contained in $\tilde{s}$. $k = 8$ gave the best results (lowest error) for forecasting the next 5 values.

The described improvement helped to remove errors generated by ANNs predicting low frequencies and to reduce time needed to train all networks.

5. An application

The described wavelet-neural method was applied to a USD/PLN average exchange rate. The archival data was downloaded from National Bank's of Poland web site (http://www.nbp.pl) and covered the period 1996.01.02 – 2002.03.08, that is 1561 values.

To test the prediction method the following procedure was developed and repeated 5 times:

1. Let $k = 100$.
2. $s = [s_1, \ldots, s_{1561-k-5}, 0, \ldots, 0]$ is a vector containing the exchange rates, zero padded to fulfill the Mallat's algorithm assumptions (length: 2^{11}).
3. 5 consecutive values: $\tilde{s}_{1561-k-4}, \ldots, \tilde{s}_{1561-k}$ are forecasted using the improved wavelet-neural method on s.
4. Predicted data is saved.
5. if $k > 1$ then $k := k - 1$ and go to step 2.
6. End.

In step 3 four MLPs were used to approximate the four highest frequency ranges, since this number of MLP forecasted ranges generated the smallest prediction error. The filter coefficients came from the Daubechies 4 wavelet family.

Optimal ANN architectures were estimated using JavaNNS (a Java interface to SNNS kernel, see [11]) and its Optimal Brain Surgeon algorithms.

The networks had an input layer (6 input neurons), one hidden layer and an output layer (1 neuron). The hidden and output neurons used the logistic activation function. Table 1 contains details about architectures and frequency ranges.

Tab. 1. Frequency ranges and ANNs architectures

Network	Range	Oscillations length	Hidden neurons
1	$\frac{f_N}{16}$ to $\frac{f_N}{8}$	16–32 samples	1
2	$\frac{f_N}{8}$ to $\frac{f_N}{4}$	8–16 samples	2
3	$\frac{f_N}{4}$ to $\frac{f_N}{2}$	4–8 samples	6
4	$\frac{f_N}{2}$ to f_N	2–4 samples	6

The MLPs were trained with the Backpropagation-momentum algorithm.

ANNs patterns were split into learning (L) and testing (T) set. The testing set contained 80 randomly selected patterns, the learning set ca. 1380–1480 (depending on k).

A typical learning result during the prediction test procedure (for a particular k) is shown in Tab. 2.

Tab. 2. A typical learning result during the prediction test procedure

Net	Set	SSE	MSE	RMSE	NRMSE	R2	DIR
1	U	0,0787	0,00005	0,0073	0,2828	0,920	85,36%
2	U	0,0717	0,00005	0,0069	0,3858	0,851	84,34%
3	U	0,0285	0,00002	0,0044	0,3391	0,884	89,85%
4	U	0,0428	0,00003	0,0054	0,4825	0,767	88,49%
1	T	0,0316	0,00040	0,0198	0,3299	0,891	90,00%
2	T	0,0051	0,00006	0,0080	0,5451	0,702	86,25%
3	T	0,0053	0,00007	0,0081	0,4577	0,790	86,25%
4	T	0,0016	0,00002	0,0044	0,3533	0,875	90,00%

Note that these are error measures computed using learning and testing patterns, but not *prediction* errors of the whole, aggregated wavelet-neural model. These are the measures definitions:

1. Sum of Square Error
$$SSE = \sum_{i=1}^{N} (y_i - \tilde{y}_i)^2.$$
2. Mean Square Error
$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \tilde{y}_i)^2.$$
3. Root of MSE
$$RMSE = \sqrt{MSE}.$$
4. Normalized RMSE
$$NRMSE = \frac{RMSE}{\sqrt{\sigma}}.$$
5. R2
$$R2 = 1 - \frac{MSE}{\sigma},$$

where $\sigma = \frac{1}{N}\sum_{i=1}^{N}(y - y_i)^2$, $y = \frac{1}{N}\sum_{i=1}^{N} y_i$. y_i, $\tilde{y}_i$ denote the expected and obtained MLP's output value on i-th pattern, respectively. DIR is the percentage of correctly predicted directions of value alteration.

After the prediction testing procedure was 5 times repeated, 2500 of predicted exchange rates were obtained. They were divided into 5 groups containing the 1st, 2nd, 3rd, 4th and 5th forecasted rate. In each of these groups all predicted values were compared to the real data to estimate the prediction error. Following error measures were used:

1. Root Average Square Error

$$RASE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(s_i - \tilde{s}_i)^2}.$$

2. Mean Absolute Percentage Error

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{s_i - \tilde{s}_i}{s_i}\right| * 100.$$

3. Theil's information coefficient

$$T^R = \frac{\sqrt{\sum_{i=1}^{R}(s_i - \tilde{s}_i)^2}}{\sqrt{\sum_{i=1}^{R}(s_i - s_{i-1})^2}},$$

where s_i is the real value, $\tilde{s}_i$ its prediction, N the number of predictions of a value.

Note that $RASE$ and $MAPE$ are applied to all $N = 500$ values in each of 5 groups, while T^R to results of each forecasting. It means that having forecasted values $\tilde{s}_{k+1}, \ldots, \tilde{s}_{k+5}$ and real data $s_{k+1}, \ldots, s_{k+5}$ five Theil's coefficients are computed:

$$T^R = \frac{\sqrt{\sum_{i=1}^{R}(s_{k+i} - \tilde{s}_{k+i})^2}}{\sqrt{\sum_{i=1}^{R}(s_{k+i} - s_{k+i-1})^2}},$$

where $R = 1, \ldots, 5$. The purpose is to focus on the relationship between a prediction's length and quality ([14], p. 87).

The forecasting method results are presented in Tab. 3.

Tab. 3. Prediction test procedure's results

	Predicted value's number				
Error measure	1	2	3	4	5
$RASE$	0,028	0,041	0,048	0,056	0,061
$MAPE$	0,544%	0,789%	0,960%	1,107%	1,227%
$Avg(T^R)$	2,281	1,363	1,492	1,670	1,808
$T^R < 1$	51,4%	33,8%	29,4%	20,8%	16,6%
$Avg(T^R < 1)$	0,322	0,240	0,211	0,155	0,131
$\sigma(T^R < 1)$	0,278	0,298	0,293	0,279	0,276
DIR	55%	58,6%	64%	51,2%	49,2%

The predicted value's number equals to R in T^R, $\sigma(.)$ denotes the standard deviation.

Values of $T^R < 1$ are exposed in Tab. 3 because of their importance. $T^R = 0$ means there was no prediction error, $T^R > 1$ means it was worse than the trivial "forecasting with the previous value".

6. Concluding remarks

The above presented results indicate that the prediction algorithm works pretty well while generating values for short time periods. The errors rise as the prognosis length is extended, which is intuitive. Simultaneously the amount of $T^R < 1$ falls. The $MAPE$, $RASE$ errors and direction coefficients from the 1st, 2nd and 3rd forecasted exchange rate are very satisfactory. The fact that DIR rises achieving the maximum value at the 3rd rate is rather surprising. This value of 64% may make some practical applications possible. However, $DIR's$ next values: 51,2% and 49,2% indicate that there is no possibility to trust the forecasted 4th and 5th value of the exchange rate direction change prognosis.

It seems that an improvement of prediction could be achieved adding some other economical data to the learning patterns (like stock market indexes, inflation rates) on which the USD/PLN exchange rate may depend.

A summary of the most important results:

1. A high (64%) direction coefficient while forecasting the future 3rd exchange rate.

2. A low (0,544%, 0,789%) $MAPE$ error while forecasting the future 1st and 2nd rates.

3. A satisfactory (51,4%) amount of good-quality (low T^R and its standard deviation) predictions of the 1st rate.

4. The designed MLPs achieved a high DIR coefficient (86,25–90%) on testing patterns.

7. Acknowledgments

I would like to thank dr Andrzej Bielecki for valuable discussions and many helpful comments on the final version of this work.

8. References

[1] Aboufadel E., Schlicker S.; *Discovering Wavelets*, John Wiley & Sons 1999.

[2] Azoff E. M.; *Monitoring Forecast Performance Using the Breakeven Locus*, Neurove$t Journal 1995, March-April, pp. 8–12.

[3] Battle G.; *A block spin construction of ondelettes. Part I: Lemarie functions*, Commun. Math. Phys. 110 (1987), pp. 601–615.

[4] Baestens D.E., Bergh van den W.M., Wood D.; *Tracking the Amsterdam Stock Index Using Neural Networks* in [19].

[5] Beltratti A., Margarita S., Terna P.; *Neural Networks for Economic and Financial Modeling*, International Thomson Computer Press, London 1996.

[6] Franklin Ph.; *A set of continues orthogonal functions*, Math. Ann. 100 (1928), pp. 522–29.

[7] Haar A.; *Zur Theorie der orthogonalen Funktionensysteme*, Math. Ann. 69 (1910), pp. 331–371.

[8] Hertz J., Krogh A., Palmer R.G.; *Introduction to the Theory of Neural Computation* Addison-Wesley Publishing Company, Massachusetts 1991.

[9] Hecht-Nielsen R.; *Kolmogorov's Mapping Neural Network Existence Theorem*, Proceedings of the International Conference on Neural Networks, Part III, IEEE, New York.

[10] Jensen A., Cour-Harbo A.; *Ripples in Mathematics. The Discrete Wavelet Transform*, Springer-Verlag Berlin Heidelberg 2001.

[11] *Java Neural Network Simulator's* homepage, http://www-ra.informatik.uni-tuebingen.de/forschung/JavaNNS.

[12] Lemarie P.G., *Ondelettes à localisation exponentielle*, J. Math. Pures Appl. 67 (1998), pp. 227–236.

[13] Lin K.; *The ABC's of BDS*, Journal of Computational Intelligence in Finance, Vol. 5, No 4, July/August.

[14] Lula P.; *Jednokierunkowe sieci neuronowe w modelowaniu zjawisk ekonomicznych*, Wydawnictwo Akademii Ekonomicznej w Krakowie, Kraków 1999.

[15] Korbicz J., Obuchowicz A., Uciński D.; *Sztuczne sieci neuronowe - podstawy i zastosowania*, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1994.

[16] Mallat S.; *Multiresolution approximation and wavelet orthonormal bases of* $L_2(\mathbb{R})$, Trans. Am. Math. Soc., 315 (1989), pp. 69–88.

[17] Osowski S.; *Sieci neuronowe w ujeciu algorytmicznym*, Wydawnictwa Naukowo-Techniczne, Warszawa 1996.

[18] Rutkowska D., Piliński M., Rutkowski L.; *Sieci neuronowe, algorytmy genetyczne i systemy rozmyte*, PWN, Warszawa–Łódź, 1997.

[19] Refenes A.P.N.; *Neural Networks in the Capital Markets*, J. Wiley & Sons, Chichester 1995.

[20] Schauder M.J.; *Einige Eigenschaften der Haarschen Orthogonalsysteme*, Math. Zeit. 28 (1928), pp. 317–320.

[21] Strömberg J.-0.; *A modified Franklin system and higher order spline systems on* $\mathbb{R}^n$ *as unconditional bases for Hardy spaces*, in: Conference in Harmonic Analysis in Honor of A. Zygmund, vol. II, Wadsworth, Belmont 1983, 475-493.

[22] Steiner M., Wittkemper H.-G.; *Neural Networks as an Alternative Stock Market Model* in [19].

[23] Tadeusiewicz R.; *Sztuczne sieci neuronowe*, Akademicka Oficyna Wydawnicza RM, Warszawa 1993.

[24] Tsibouris G., Zeidenberg M.; *Testing the Efficient Markets Hypothesis with Gradient Descent Algorithm* in [19].

[25] White H.; *Economic Prediction Using Neural Networks: The Case of IBM Daily Stock Returns*, Proceedings of the IEEE International Conference of Neural Networks, San Diego 1988.

[26] Witkowska D.; *Sztuczne sieci neuronowe i metody statystyczne. Wybrane zagadnienia finansowe*, Wydawnictwo C.H. Beck, Warszawa 2002.

[27] Wojtaszczyk P.; *A Mathematical Introduction to Wavelets.*, Cambridge University Press 1997.

[28] Zirilli J.S.; *Financial Prediction Using Neural Networks*, International Thomson Computer Press, London 1966.

[29] Żurada J.; *Introduction to Artificial Neural Systems*, PWS Publishing Company 1992.

Received June 20, 2002